Molecular Dynamics for Proteins: Performance Evaluation on Massively Parallel Computers Based on Mesh Networks Using a Space Decomposition Approach

P. BALLESTRERO, P. BAGLIETTO, and C. RUGGIERO*

Department of Communication, Computer, and Systems Sciences, University of Genoa, Via Opera Pia, 13, I 16145 Genoa, Italy

Received 29 November 1994; accepted 19 June 1995

ABSTRACT

Parallel computing seems to be the solution for molecular dynamics of large atomic systems, such as proteins in water environments, but the simulation time critically depends on the processor allocation strategy. A study of the optimal processor allocation based on a space decomposition algorithm for single instruction multiple data flow mesh computers is presented. A particular effort has been made to identify the best criterion according to which the atoms can be allocated to the processors using a spatial decomposition approach. The computing time depends on the granularity of the space decomposition among processing elements and on the ratio between the computation power of processing elements and the communication speed of the interprocessor network.

Introduction

Molecular modeling and computer simulation are essential tools in modern biophysics research because of their complementary features with respect to experimental approaches (e.g., X-ray, neutron scattering spectroscopy, nuclear magnetic resonance [NMR] spectroscopy) and because they allow investigation of the spatial features of a molecular system, thus saving time and effort.

The computer simulation area involves several computational methods, such as Monte Carlo simulation, energy minimization, and molecular dynamics (MD). These computational approaches require solution of the potential energy function of a molecular system (MD also adds a kinetic energy term) and belong to the research field called molecular mechanics (MM). They are limited by the computational power of today's computers and by the accuracy of the mathematical models em-

* Author to whom all correspondence should be addressed.

CCC 0192-8651 / 96 / 040469-07

ployed.[1] Computer simulations use as starting point a low-energy conformation of the system in order to compute its structural and thermodynamic properties. Proteins and, more generally, biological molecules involve additional conceptual and practical problems with respect to crystal structures, mainly due to their amorphous structure and to the presence of the solvent. Moreover, because of the low dielectric constant (2–4) in the protein core, the long-range electrostatic interactions cannot be neglected.

Since MM simulations are highly computer intensive and have a high degree of intrinsic parallelism, the use of parallel computers would seem appropriate. Specifically, single instruction multiple data flow (SIMD) computers are well suited to MM simulation tasks.[2] A SIMD machine is a computer with thousands of equal processing elements (PEs) executing the same instruction flow on different sets of data and interacting with each other by means of a high-speed communication network. Each PE has its own memory for data storage and can exchange data with the other PEs simply by message passing. Since the time required for communication increases with the distance of the interacting PEs, a key feature for an efficient parallel algorithm is matching between the pattern of communications required by the algorithm and the PE interconnection network.

Since MM simulation algorithms are based on a 3D data structure, a simple 3D or 2 D mesh interconnection network fits well with the interprocessor communication pattern usually required by such algorithms.[3, 4]

We present a CHARMM-like[5] data parallel algorithm for MD suitable for implementation on massively parallel computers with a 3D or 2D mesh interconnection network, in which each unit communicates directly with six neighbors: North, East, South, West, Up, and Down (in 2D meshes each PE can communicate with four or six neighbors; see, for example, the Distributed Array Processor (DAP) interprocessor network and the Mas Par X-network). This algorithm exploits the intrinsic parallelism of the simulation to reduce the turnaround time starting from a space decomposition data structure.

Methods

PROTEIN POTENTIAL ENERGY FUNCTION

The principle of MD is numerical integration of the classical equations of motion for a system of interacting particles. In protein dynamics simulations, the particles involved are the protein atoms, interacting by covalent bonds and nonbond interactions, and the solvent atoms. The force acting on each atom can be computed by taking the derivative of the potential energy function with respect to its position. Let i be the atom number ($1 < i < N$, where N is the total number of atoms), let $F_i^j(t)$ be the scalar force acting on the atom i in the direction j (j: x, y, x), and let be $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$, the unit vectors of the simulation space. Equation (1a) states the total vector force acting on the atom i.

$$\mathbf{F}_i(t) = F_i^x(t)\mathbf{e}_x + F_i^y(t)\mathbf{e}_y + F_i^z(t)\mathbf{e}_z \quad \text{(1a)}$$

Let $V(x_1, y_1, z_1, \ldots x_N, y_N, z_N)$ be the total potential energy as a function of all the particle positions; the force components acting on the generic particle i are

$$F_i^x(t) = -\frac{\partial}{\partial x_i} V(x_1, y_1, z_1, \ldots, x_N, y_N, z_N)$$

$$F_i^y(t) = -\frac{\partial}{\partial y_i} V(x_1, y_1, z_1, \ldots, x_N, y_N, z_N)$$

$$F_i^z(t) = -\frac{\partial}{\partial z_i} V(x_1, y_1, z_1, \ldots, x_N, y_N, z_N) \quad \text{(1b)}$$

Regarding the second-order series of Taylor as an acceptable approximation, the motion equations can be written as follows:

$$\mathbf{r}_i((n+1)\Delta T) = \mathbf{r}_i(n\Delta T) + \Delta T\,\boldsymbol{v}_i(n\Delta T) + \tfrac{1}{2}\Delta T^2\mathbf{a}_i(n\Delta T)$$

$$\boldsymbol{v}_i((n+1)\Delta T) = \boldsymbol{v}_i(n\Delta T) + \frac{\Delta T}{2}[\mathbf{a}_i(n\Delta T) + \mathbf{a}_i((n+1)\Delta T)] \quad \text{(2)}$$

where ΔT is the integration step, $n\Delta T$ is the current time, and $(n+1)\Delta T$ is the time of the new integration step; i is the number of the current atom and runs from one to N (the maximum atom number). Equation (2) represents the velocity version of the Verlet algorithm.[6, 7] It is used to compute the position and the velocity of each atom at time $\Delta(n+1)T$, given their present values.

Other integration techniques, such as Runge-Kutta methods, require several force evaluations per timestep. Thus they are unsuitable for molecular dynamics of proteins, where force evaluation is the most time-consuming part of the simulation.[8]

We adopted the protein force fields used by CHARMM, which is one of many similar methods

for performing molecular mechanics; however, the algorithm can be used for other force fields.

$$V(r_1, r_2, r_3, r_4, \ldots, r_N) = \sum_{i=1}^{B} \tfrac{1}{2}K_{bi}(b_i - b_{0_i})^2 + \sum_{i=1}^{A} \tfrac{1}{2}K_{\alpha_i}(\alpha_i - \alpha_{0_i})^2 + \sum_{i=1}^{\Phi} K_{\phi_i}[1 + \cos(n_i\phi_i - \phi_{0_i})] + \sum_{i=1}^{\Xi} \tfrac{1}{2}K_{\xi_i}(\xi_i - \xi_{0_i})^2 + \sum_{0<i<j}^{N} \left[\frac{A_{ij}}{r_{ij}^{12}} + \frac{C_{ij}}{r_{ij}^{6}} + \frac{Q_iQ_j}{4\pi\varepsilon_0\varepsilon_r r_{ij}}\right] \quad (3)$$

In eq. (3), b, b_0, and K_b are the bond length, its equilibrium value parameter, and the bond stretching force constant, respectively; α, α_0, and k_α are the bond angle, its equilibrium value parameter, and the angle bending force constant, respectively; ϕ, k_ϕ, n, and ϕ_0 are a dihedral angle, its force constant, the symmetric multiplicity, and phase, respectively; ξ, k_ξ, and ξ_0 are the improper dihedral angle, its force constant, and its equilibrium value parameter, respectively; Q_1 and Q_2 are the electrostatic charges of the two atoms; r is the distance between them; A_{ij} and C_{ij} are the coefficients of the 6–12 van der Waals potential for the atoms i and j; B is the bond number; A is the angle number; Φ is the dihedral angle number; Z is the number of improper dihedral angles; and N is the atom number.

DATA STRUCTURE

The performance of an algorithm implemented on a SIMD massively parallel computer depends on the way in which the data is distributed among processors. Computations on a processor's local memory are relatively fast with respect to data communication between PEs which are not close to each other. Basically, there are three different strategies in assigning data to PEs using a data parallel molecular dynamics approach:

- Assignment of a fixed set of particles to a particular PE (atom decomposition)
- Assignment of a fixed set of interactions to a particular PE (force decomposition)
- Assignment of small fractions of the simulated space to PEs (space decomposition).

In SIMD computers, every PE must execute the same instruction at each simulation step, so that the more balanced the computational charge among PEs, the higher the hardware exploitation. The atom and force decomposition strategies present the advantage of using a more balanced data structure with respect to the space decomposition strategy.[10] On the other hand, the data structure used with the atom and force decomposition strategies is strictly coupled to the hardware architecture. Each PE has to take care of a fixed set of atoms or interactions, but during the simulation process some particles, especially water molecules, move around in the simulation space so that the set of nonbound interactions changes, causing enormous irregularity in the communication algorithm among nearby PEs. Moreover, when simulating large systems, such as proteins in water solutions, a cutoff limit in computing the nonbond interactions is necessary to reduce the simulation time. This gives rise to enormous problems[11] because SIMD computers are best exploited if all the interactions are treated identically. The space decomposition strategy is a solution to these problems because it keeps the data elements (protein and water atoms) completely decoupled from computer architecture and allows atoms to move from one PE to another, and the cutoff is treated in a simple and efficient manner (see the cutoff sphere and the communication algorithm in Fig. 2). Moreover, using a space decomposition strategy on 2D and 3D architectures with torus facilities, the problem of the periodic boundary condition is treated automatically, without computation overhead. The main difficulty in obtaining an optimal PE space allocation is due to the irregular density of the proteins, which causes an unbalanced distribution of the computational charge. Here simulation performances are considered as functions of the spatial atomic density of globular proteins and of the granularity of the space partitioning among PEs.

ALGORITHM

We have divided the total simulated space, containing one or more proteins, and the aqueous environment into cubic elements of fixed side called voxels. The data structure used by the MD simulator is based on the association among these voxels and the PEs. The association changes with the computer size and the space partitioning model. If the PE number is equal to the voxel number, one PE works on one voxel; otherwise, if the PE number is smaller than the voxel number,

one PE must be shared by a set of voxels. This change of the the processor–voxel association can be taken into account without loss of generality by introducing a virtual machine in which each processor (virtual processing element, VPE) is associated with one voxel. Introducing the virtual machine is also useful in comparing the potentiality of 2D and 3D meshes, because each PE of a square mesh can be associated with a column of VPE, transforming a real 2D computer into a virtual cube. A generic VPE i computes the interactions acting on the atom j if the center of the sphere representing it is inside voxel i (i.e., the voxel associated with the VPE i). Our approach does not take into account the case with more PEs than voxels, because associating one voxel to a set of PEs is equivalent to dividing the voxel into smaller subvoxels.

The algorithm includes two fundamental different actions that are iterated at each integration step: (1) interaction computing, and (2) atom moving. Interaction computing consists of a sequence of communication actions and floating point operations which update the forces acting on each atom. This is the most computer-intensive action because it involves the movement of a large quantity of bytes through the interprocessor network and force computation.

Atom moving involves communication among nearby VPEs only, in order to move all atoms that must be shifted from one voxel to another. In the MD simulation, we used the atom and amino acid parameters of CHARMM (version 2.1). To speed up the communication between VPEs, we adopted an optimized data structure using a number of bits as low as possible. This data structure is set dynamically on the basis of the molecular system simulated. If we run a simulation on a protein containing the 20 standard L-amino acids only, for each atom the VPEs send through the network the data element represented in Figure 1. If metal atoms or other chemical groups are also present, the size of the data structure must increase, including more bits to code the new atoms. The fields M (Mass), C (electrostatic charge), and K (atom kind) are coded using a number of bits as low as possible. Number is the atom index, and X, Y, Z are the atom coordinates. The M field is 4 bits long, so 16 values can be coded—each PE memory contains a lookup table with the real mass values. This allows us to send 4 bits only for M through the network. The same applies to field C (5 bits) and field K (5 bits).

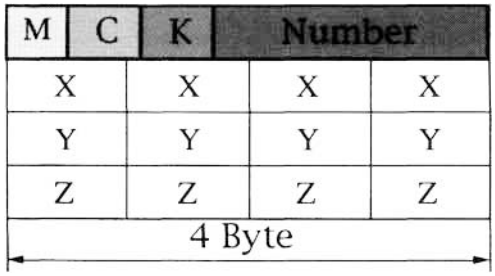

FIGURE 1. A schematic representation of the atom data structure (used for proteins made only of the 20 amino acids, CHARMM parameters): M is the atom mass (4 bits, 16 different values); C is the atom charge (5 bits, 32 different values); K is the atom kind (5 bits, 32 different values); X, Y, and Z are the spatial coordinates.

Figure 2 shows a 2D schematic representation of the communication algorithm adopted to compute the nonbond interactions inside the cutoff sphere.[12,13] We call the smaller cube of voxels containing the cutoff sphere the cutoff cube. To compare the nonbond interactions inside the cutoff sphere, the central VPE has to compute all forces among the atoms inside the central voxel and all intervoxel interactions with the atoms inside the voxels connected by the arrow loop (Fig. 2a). This means that the central processor has to compute only a half of all interactions among the atoms inside the cutoff sphere. The remaining part of the nonbond interactions is computed by the neighbor VPEs during their interaction computing cycle (Figs. 2b–e). Figure 3 shows the 3D communication algorithm. As in the 2D representation, the central processor (number 0) communicates only with half of the PEs inside its cutoff sphere.

The larger the cutoff radius, the larger the number of communication actions and force computations per step. In out theoretical study and in the simulation on the MASPAR MP-2, we have used a cutoff radius of 8 Å for van der Waals interactions and a radius of 15 Å for electrostatic interactions.[14] Let the following variables be defined:

- T: time taken by a simulation step
- T_{ce}: time taken to compute the electrostatic interaction between two atoms
- T_{cw}: time taken to compute the van der Waals interaction between two atoms
- T_{cb}: time taken to compute all bond interactions
- T_{sr}: time taken to transfer an atom data structure between two nearby PEs

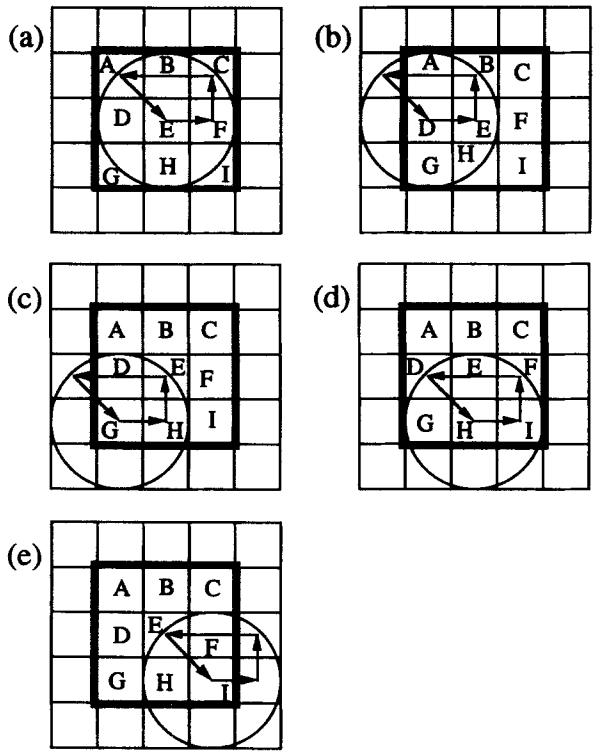

FIGURE 2. A 2D representation of the communication algorithm. The central processor communicates only with the other processing elements within its cutoff cube connected by the arrow loop. (a) The processing element associated with the voxel E computes the interactions among its atoms and the atoms with voxels A, B, C, F. (b–e) At the same time, the processing elements associated with the voxels D, G, H, I computes the cutoff sphere centered in E, computing the interactions D — E, G — E, H — E, and I — E.

- n: maximum number of atoms inside a voxel (in a SIMD machine, all PEs have to wait for the PE working in the largest set of atoms)
- D_e: the diameter of the cutoff sphere for electrostatic interactions, expressed as number of voxels

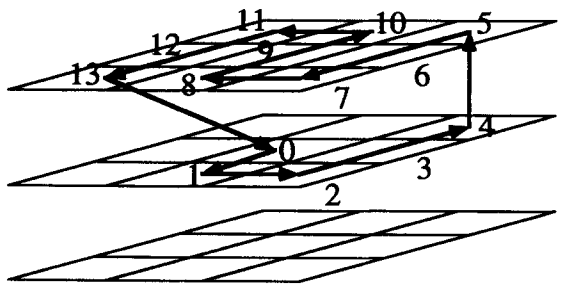

FIGURE 3. A 3D representation of the communication algorithm.

- D_w: the diameter of the cutoff sphere for van der Waals interactions, expressed as a number of voxels
- M_v: the number of VPEs
- M_r: the number of (real) PEs

At each simulation step, the central VPE has to interact with $0.5(D_e^3 - 1)$ neighbors, so the total communication time is $\leq 0.5n(D_e^3 - 1)T_{sr}$. When computing the electrostatic interactions, the central processor has to operate on the atoms inside $1 + 0.5(D_e^3 - 1)$ voxels (this is also true when computing the van der Waals interactions, but with a different cutoff radius of 8 Å).

The total time taken by a simulation step is

$$T = \frac{M_v}{M_r}\left\{\frac{n}{2}(D_e^3 - 1)T_{sr} + \frac{n^2}{2} \times \left[(D_e^3 - 1)T_{ce} + (D_w^3 + 1)T_{cw}\right] + T_{cb}\right\} \quad (4)$$

The scale factor M_v/M_r converts the computation time of the virtual machine into the real computation time. If M_v is equal to M_r, then a real processor works on the atoms of one voxel, and the virtual and the real computers have the same number of PEs.

Results and Conclusions

The D_e factors of eq. (4) are the most computer-expensive ones because of the long-range feature of the electrostatic forces, which requests a large cutoff limit. The time T required to compute the forces acting on each atom can be approximated by eq. (5):

$$T \cong \frac{M_v}{M_r}\left[\frac{n}{2}(D_e^3 - 1)T_{sr} + \frac{n^2}{2}(D_e^3 + 1)T_{ce}\right] \quad (5)$$

The dashed curve in Figure 4 shows n (the maximum number of atoms within a voxel) as a function of the size of the voxel side for globular proteins. In our theoretical study and in our MD simulations on MASPAR MP-2, we have considered the structure of Exotoxin-A[15] (from *Pseudomonas aeruginosa*). Similar results have been obtained using Ribonuclease-A[16] and other globular proteins with similar size. The continuous curve represents the relation between the dimension (s) of the voxels and D_e.

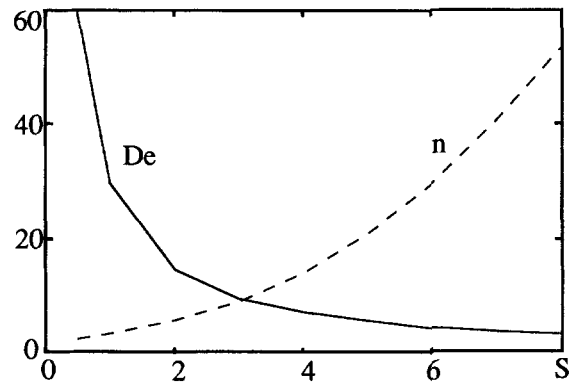

FIGURE 4. The dashed curve represents the maximum number of atoms (n) inside a voxel with side ranging from 0.5 Å to 8 Å for *Pseudomonas aeruginosa*; the continuous curve is the relation between the voxel linear dimension (s) and the number of voxel per side of a cutoff cube (D_e) with an electrostatic cutoff limit of 15 Å.

Figure 5 shows the theoretical value of T (the simulation time for an integration step) as a function of the chosen voxel side for 2D meshes such as MASPAR MP-2 (T_{sr} = 20 μs, T_{ce} = 390 μs), DAP 610-C (T_{sr} = 10 μs, T_{ce} = 250 μs), and a computer model with $T_{sr} = T_{ce}$ = 130 μs. The MASPAR MP-2 is a SIMD computer based on a square mesh network. Its number of PEs ranges from 1024 to 16384. Each PE is a simple arithmetic logic unit (ALU) without floating point processor. Its peak performance is about 2400 MFLOPS (16,384 PE-64-bit floating point operations). The DAP 610-C is similar to MASPAR MP-2, but each PE has an 8-bit floating point processor that improves the computation performance.

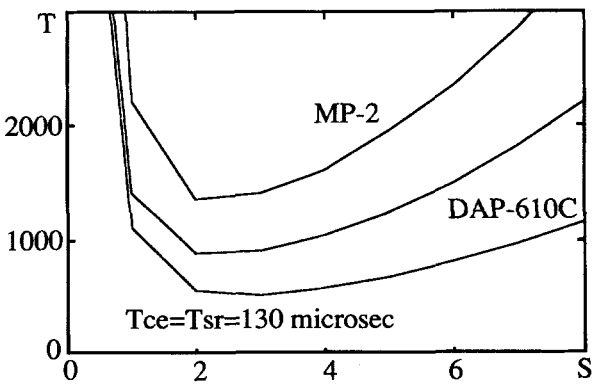

FIGURE 5. The relations between the time T (seconds) taken for one integration step and the space partitioning are shown for MASPAR MP-2, DAP 610-C, and a low-granularity mesh with faster processing elements and slower network ($T_{sr} = T_{ce}$ = 130 μs).

The choice of 130 μs [0.5*(10 μs + 250 μs)]—the average of the DAP 610-C time values—derives from a general rule of the VLSI system design, which states that the communication speed of an electrical line (communication channel) inside electronic chips, or connecting different chips, is a function of its physical length. The closer the transmitter and receiver, the lower the effects of the electrical parasite capacity and the lower the communication time (T_{sr}). This means that using bigger and more powerful PEs increases the communication time.[17,18] So 130 μs is a straightforward choice for the voxel side in extrapolating from the values of T_{sr} and T_{ce} of real computers.

The main difference between 2D and 3D computing concerns the association among PEs and columns of voxels. With square meshes, a PE must compute the interaction acting on the atoms of a voxel column; whereas when using 3D computers, if the space partitioning is not too fine, it is possible to associate one PE with one voxel. Figure 6 shows the time taken by a simulation step using 3D architectures. The lower curve refers to PEs with the same time values of the MASPAR MP-2; the upper curve refers to PEs with $T_{sr} = T_{ce}$ = 205 μs—the average value of the MASPAR MP-2. We have chosen MASPAR MP-2 as a template for 3D meshes because its PEs do not include floating point processor and are smaller and simpler than the DAP 610-C PEs. In other words, a cubic MASPAR-like mesh is a more realistic model than a cubic mesh made of the more complex DAP PEs.

In Figure 7 the theoretical T values and the real values of the simulation of Exotoxin-A on MASPAR MP-2 are compared, showing that the course of the time T in simulation is in agreement

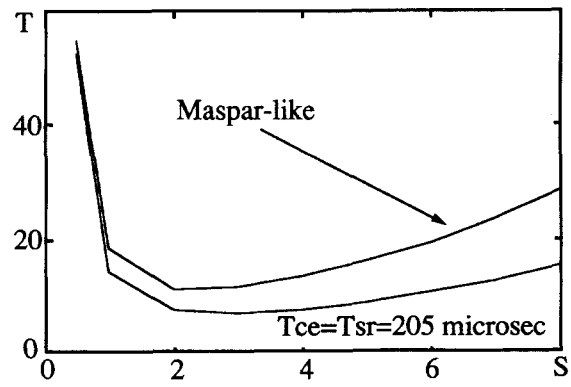

FIGURE 6. The relations between the time T (seconds) taken for one integration step and the space partitioning are shown for 3D computers.

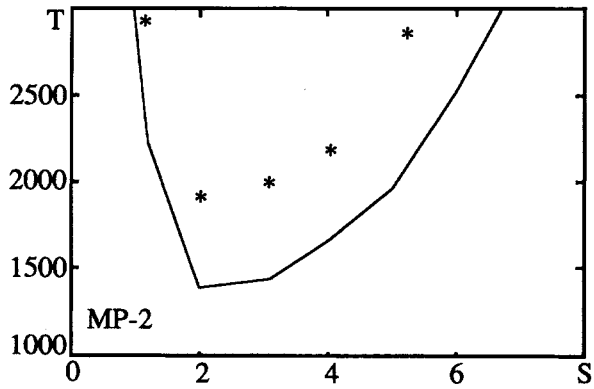

FIGURE 7. Comparison between the theoretical and simulation values of T for MASPAR MP-2.

with the theoretical one. The differences derive from the time needed to decode the atom information (using the distributed lookup tables), which has been neglected in the theoretical analysis.

In the theoretical cases which have been considered here and in the real simulation on MASPAR MP-2, when s decreases and reaches a value below 2 Å the value of T rapidly increases because the atom distribution among PEs becomes more unbalanced. In this case many PEs have no atoms in their voxels and waste time waiting for the other PEs. Moreover, given the distance between two interacting atoms, when the voxel side decreases, the PE number involved in the communication algorithm increases, so the number of message passing steps (to compute the interactions between the two atoms) increases.

The computational performance depends on the T_{sr}/T_{ce} value because the communication costs depend on n linearly, while the computation for pairwise interactions has quadratic dependence. The greater the T_{sr}/T_{ce} value, the greater the linear term in eqs. (4) and (5) and the shorter the computation time.

References

1. S. L. Lin, J. Mellor-Crummey, B. P. Pettitt, and G. N. Phillips, *J. Comp. Chem.*, **13**, 1022 (1992).
2. D. Finchman and N. Quirke, *Information Quarterly for MD and MC Simulations* (the CCP5 Newsletter), **10**, 13 (1993).
3. Y. Feldman and E. Shapiro, *Commun. ACM*, **35**, 61 (1992).
4. T. Hoshino, *PAX Computer—High Speed Parallel Processing and Scientific Computing*, Addison-Wesley, Reading, MA, 1985, p. 80.
5. B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus, *J. Comp. Chem.*, **4**, 187 (1983).
6. L. Verlet, *Phys. Rev.*, **159**, 98 (1967).
7. M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*, Claredon Press, Oxford, 1987, p. 78.
8. J. M. Goodfellow, *Molecular Dynamics, Applications in Molecular Biology*, Macmillan, London, 1991, p. 8.
9. D. Fincham, *Molec. Simulation*, **1**, 1 (1987).
10. S. Plimpton, B. Hendrickson, and G. Heffelfinger, In *Proc. of 6th Conference on Parallel Processing for Scientific Computing*, Norfolk, VA, 1993. Richard F. Sincovec et al., Eds., Society for Industrial and Applied Mathematics, Philadelphia, p. 178.
11. D. Fincham and B. J. Raltson, *Comput. Phys. Commun.*, **23**, 165 (1981).
12. W. Form, N. Ito, and G. A. Kohring, *Internat. J. Modern Phys.*, **4**, 6 (1993).
13. D. M. Beazley and P. S. Lomdahl, *Parallel Comput.*, **20**, 173 (1994).
14. J. M. Goodfellow, *Molecular Dynamics, Applications in Molecular Biology*, MacMillan, London, 1991, p. 13.
15. R. J. Collier and D. B. McKay, *J. Mol. Biol.*, **157**, 413 (1982).
16. A. Wlodawer, L. A. Svensson, L. Sjolin, and G. L. Gilliland, *Biochemistry*, **27**, 2705 (1988).
17. N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A System Perspective*, Addison-Wesley, Reading, MA, 1988.
18. J. R. Nickolls, Interconnection Architecture and Packaging in Massively Parallel Computers, Proceedings of the Packaging, Interconnects and Optoelectronics for the Design of Parallel Computers Workshop, IEEE/LEOS, 1992.